A ROBUST SOLUTION PROCEDURE FOR HYPERELASTIC SOLIDS WITH LARGE BOUNDARY DEFORMATION*

Suzanne M. Shontz† Stephen A. Vavasis‡

†Department of Computer Science and Engineering
The Pennsylvania State University, University Park, PA 16802
shontz@cse.psu.edu.

‡Department of Computer Science
Cornell University, Ithaca, NY 14853
vavasis@cs.cornell.edu.

Abstract

We propose a solution procedure for the finite element discretization of a static nonlinear compressible Mooney-Rivlin hyperelastic solid. We consider the case in which the boundary condition is a large prescribed deformation, so that mesh tangling becomes an obstacle for straightforward algorithms. Our solution procedure involves a largely geometric procedure to untangle the mesh: solution of a sequence of linear systems to obtain initial guesses for interior nodal positions for which no element is inverted. After the mesh is untangled, we take Newton iterations to converge to a mechanical equilibrium. The Newton iterations are safeguarded by a line search similar to one used in

*Much of the work of the first author was performed while a graduate student in the Center for Applied Mathematics at Cornell University. The author's work is supported by the National Physical Science Consortium, Sandia National Laboratories, and Cornell University. The work of both authors is supported in part by NSF grant ACI-0085969.

optimization. Our computational results indicate that the algorithm is much faster than a straightforward Newton continuation procedure and is also more robust (i.e., able to tolerate much larger deformations).

1 The problem under consideration

We consider the problem of solving for the deformed shape of a hyperelastic solid body under static loads. The continuum mechanical model under consideration has the following description [6]. Let $B_0 \subset \mathbf{R}^d$ be an undeformed solid body whose boundary is ∂B_0. Here d, the space dimension, is 2 or 3. Assume boundary conditions (either displacement or traction, i.e., Dirichlet or Neumann) are given as follows. The boundary ∂B_0 is partitioned into two subsets Γ_D and Γ_N. A function $\phi_0 : \Gamma_D \to \mathbf{R}^d$ specifies new-position (Dirichlet) boundary conditions. A second function $\mathbf{t}_0 : \Gamma_N \to \mathbf{R}^d$ specifies traction (Neumann) boundary conditions. Everything in this paper extends to the more general case that some coordinate entries are Neumann while others are Dirichlet at certain boundary points, but we limit the discussion to the special case that each boundary point is Dirichlet or Neumann in all d coordinates in order to simplify notation. Finally, the model requires a specification of the model's body forces, that is, a function $\mathbf{b} : B_0 \to \mathbf{R}^d$ that specifies the force of gravity and other forces on the body.

The problem is to find a function $\phi : B_0 \to R^d$ that specifies the new position of the body. Let B denote $\phi(B_0)$. For a point $\mathbf{X} \in B_0$, let $\mathbf{x} = \phi(\mathbf{X})$. Let F be the deformation gradient, i.e., $F = d\phi/d\mathbf{X} = d\mathbf{x}/d\mathbf{X}$. It is assumed that $F(\mathbf{X})$ has a positive determinant for all $\mathbf{X}$. The *Green strain* is defined to be $E = (F^T F - I)/2$. Let scalar function $\Psi(F)$ be the *strain energy function*, which is assumed to be a property of the material. For this paper, we assume that Ψ depends only on two matrix invariants of E, namely $J = \det(F) = \sqrt{\det(2E + I)}$ and $I_1 = \mathrm{trace}(F^T F) = \mathrm{trace}(2E + I)$. Further specializing the model, the strain energy is then taken to have the following form suggested by Ciarlet and Geymonat in [2] for compressible Mooney-Rivlin materials

$$\Psi(F) = \frac{\lambda}{4}(J^2 - 1) - \left(\frac{\lambda}{2} + \mu\right) \ln J + \frac{\mu}{2}(I_1 - 3) \tag{1}$$

where $\lambda, \mu > 0$ are material parameters. For the $d = 2$ case, we assume that B_0 is 3D but that the z-displacement is identically 0 and that the x- and

y-displacements depend only on x and y; these are called *plane strain* assumptions. Thus, E has a last row and column of all zeros, and the Mooney-Rivlin formula in (1) is applied to this E to come up with the strain energy function for the $d = 2$ case. The condition for static equilibrium (written in minimization form) is that

$$\int_{B_0} \Psi(F(\mathbf{X}))\, dV - \int_{B_0} \rho \mathbf{b} \cdot \phi(\mathbf{X})\, dV - \int_{\Gamma_N} \mathbf{t}_0 \cdot \phi(\mathbf{X})\, dA \tag{2}$$

is minimized among all choices of ϕ that satisfy the Dirichlet boundary condition, i.e., that satisfy $\phi(\mathbf{X}) = \phi_0(\mathbf{X})$ for all $\mathbf{X} \in \Gamma_D$.

This condition can be rewritten in variational form: for all admissible variations $\delta\mathbf{u}$, that is, functions in the space $[H^1(B_0)]^d$ that vanish on Γ_D,

$$\int_{B_0} \frac{\partial \Psi}{\partial F} : \text{Grad } \delta\mathbf{u}\ dV - \int_{B_0} \rho \mathbf{b} \cdot \delta\mathbf{u}\, dV - \int_{\Gamma_N} \mathbf{t}_0 \cdot \delta\mathbf{u}\, dA = 0, \tag{3}$$

where $A : B = \text{trace}(AB^T)$ is used to denote the inner product of second-order tensors A and B. This model also applies to the case of linear elasticity with two changes in definitions. First, $E = ((F - I)^T + (F - I))/2$ in the case of linear elasticity. Second, $\Psi = \mu \sum_{i,j} E(i,j)^2 + \frac{\lambda}{2} \left(\sum_i E(i,i)\right)^2$, which can be written in terms of the two matrix invariants of E.

We should mention that our method does not appear to depend so much on the specific details of the Mooney-Rivlin model, except for the $\ln J$ term, which is quite important for our analysis. Since $dv = J\, dV$, where dv is the volume element of B and dV is the volume element of B_0, this logarithmic term resists infinite compression of the material: if a small positive volume of material in B_0 shrinks to a 0-volume set in B, then this term causes the strain energy at those points to become infinite.

We next describe the discretization of the problem under consideration [1]. We assume that B_0 is discretized with a mesh of triangles or tetrahedra. We assume that the discretization of ϕ, or alternatively the discretization of the displacement $\mathbf{u} = \phi(\mathbf{X}) - \mathbf{X}$, is piecewise linear, with the pieces of linearity being the mesh cells. (In Section 8, we discuss extension of our method to piecewise quadratic displacements.) This assumption implies that $\mathbf{u}$ is determined by dm real numbers, namely, the values of $\mathbf{u}$ at nodes. Recall that d is the space dimension, and let m denote the number of non-Dirichlet nodes of the mesh. The finite element method finds the displacement $\mathbf{u}$ such that (3) holds for all test functions $\delta\mathbf{u}$ in the test function space. Here, the

test function space is the set of $\delta\mathbf{u}$'s that are piecewise linear and continuous and vanish on Γ_D. The integral in (3) is evaluated with a quadrature rule; we have used a 6-point formula having degree 4 precision from [7] for our quadrature in 2D and a 15-point formula having degree 5 precision from [8] for 3D. These quadrature rules were chosen because they have positive weights; the Gauss points are interior; and they possess full triangular or tetrahedral symmetry, respectively. It suffices to solve (3) for the dm choices of $\delta\mathbf{u}$ that compose the standard basis for the test function space. This yields a system of dm nonlinear equations for dm unknowns.

The algorithmic question under consideration is how to robustly solve these nonlinear equations. In the next section, we give a summary of the mesh tangling issue and of our proposal to overcome it. The individual steps of our algorithm are then described in more detail in Sections 4 and 5. Our computational experiments are presented in Sections 6 and 7. Concluding remarks are presented in Section 8.

2 Mesh tangling

The standard method for solving a system of nonlinear equations is Newton iteration. It is well-known, however, that if the initial guess is far from the true solution, then Newton iteration will often diverge.

In the case of hyperelasticity with large deformation, there is a specific obstacle that may cause divergence, namely, mesh tangling. The definition of this term is that a mesh is *tangled* if the value of J defined in the previous section is 0 or negative in B_0. In the case of linear displacements, J is piecewise constant, and hence this condition can be verified with a finite number of determinant computations. The matter of checking for tangling in the piecewise quadratic case is more complicated and is discussed in Section 8. A solution with a tangled mesh is physically invalid. Indeed, the strain energy function is undefined in this case because of the presence of the term $-\left(\frac{\lambda}{2}+\mu\right)\ln J$. Note that although the strain energy function is undefined when J is negative, the Galerkin form (3) is still well defined, which is an anomaly that we return to below. We assume that the given problem instance has a valid solution, i.e., there is a piecewise linear function $\mathbf{u}$ satisfying the boundary conditions, as well as (3) for all test functions $\delta\mathbf{u}$ plus the condition that $J > 0$ on every element.

Even with this assumption, Newton's method will still often run into

problems because the mesh will become tangled on intermediate steps. For example, the starting point for Newton's method is often taken to be $\mathbf{u} = \mathbf{0}$ on every interior node. If the deformation of the boundary is large, then this starting point will have tangling among most of the elements that are adjacent to the boundary.

To understand a difficulty posed by a tangled mesh, suppose that the strain energy has a single term $\Psi(J) = -\left(\frac{\lambda}{2} + \mu\right) \ln J$ (for $\frac{\lambda}{2} + \mu > 0$) on a single element and that there are no boundary constraints. If we treat J as the independent scalar variable, then Newton's method for minimizing this scalar function is

$$J^{(i+1)} = J^{(i)} - \Psi'(J^{(i)})/\Psi''(J^{(i)})$$

which simplifies in this case to

$$J^{(i+1)} = 2J^{(i)}.$$

For positive $J^{(0)}$, this iteration produces a sequence of J's tending to $+\infty$. This is to be expected since the minimum of $-\left(\frac{\lambda}{2} + \mu\right) \ln J$ is indeed at $+\infty$. On the other hand, for a negative $J^{(0)}$, this iteration tends to $-\infty$, which is physically invalid.

The preceding analysis, although naive, seems to point to the following conclusion: Newton's method on the Galerkin form, when applied to a tangled mesh, has a natural tendency to make the tangling worse. We suspect that this fact is probably already known to experts in the field, although we have not been able to find it in the previous literature.

Given the conclusion in the previous paragraph, it seems of paramount importance to avoid tangling. When Newton's method fails in computational mechanics, it is standard practice to try Newton continuation, that is, to apply the load in incremental steps and use the converged solution for one step as the Newton starting point for the next step. Continuation is described in more detail in Section 3. Continuation, however, addresses the tangling issue only in an indirect fashion and therefore is likely to be very inefficient. Our computational experiments confirm the inefficiency of continuation.

We propose a new algorithm for getting around the mesh tangling obstacle. The basic idea is to first untangle the mesh using a much simplified mechanical model. Once the mesh is untangled, the true mechanical model is solved. "Untangling the mesh" means finding a ϕ that satisfies the Dirichlet boundary condition and also satisfies $J > 0$. Our new algorithm, which we call UBN (for "untangling before Newton") consists of three steps.

1. First, we attempt to untangle the mesh with the iterative-stiffening algorithm, described in Section 4. Iterative stiffening builds on the FEMWARP algorithm from our previous work [16]. If the iterative stiffening algorithm cannot untangle the mesh, then UBN reports failure to solve the problem.

2. Else if iterative stiffening succeeds, then we take Newton iterations to solve (3). The starting point for Newton is an untangled mesh produced by step 1. No continuation is used. On the other hand, Newton's method is safeguarded using a line search described in Section 5, which prevents the introduction of new tangling. The line search is based on a technique common in the interior point literature.

3 Newton continuation

In the case that direct use of Newton's method to find ϕ fails to converge, the standard alternative is Newton continuation, also known as applying the load in steps. This section briefly describes Newton continuation before we return to a description of UBN. Newton continuation is also used to make graphs of load-displacement relationships, but in this paper we assume that the problem under consideration is to determine a single final configuration rather than an entire loading path.

The basic form of Newton continuation is quite straightforward: a sequence of parameters $0 = \lambda_0 < \lambda_1 < \cdots < \lambda_N = 1$ is chosen, and a sequence of displacement vectors $\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_N$ is computed, in which for each k, $\mathbf{u}_k$ is the solution to the discretized (3) in the case that $(\phi_0(\mathbf{X}) - \mathbf{X}, \mathbf{b}, \mathbf{t}_0)$ are replaced by $\lambda_k \cdot (\phi_0(\mathbf{X}) - \mathbf{X}, \mathbf{b}, \mathbf{t}_0)$. Solution $\mathbf{u}_0$ (corresponding to absence of loads) is identically $\mathbf{0}$. Solution $\mathbf{u}_k$ is found via Newton's method, where $\mathbf{u}_{k-1}$ is used as the initial guess. The final deformed configuration is given by $\mathbf{u}_N$ since $\lambda_N = 1$. Note also that it is possible to accept a low-accuracy (not fully converged) solution for $\mathbf{u}_k$ when $k < N$ since it is presumably not necessary to achieve high accuracy for intermediate results that are not part of the ultimate answer.

In some cases, a straight linear parametrization of the load path (as in the previous paragraph) is not feasible. In this case, one must construct a nonlinear parametrization $(\phi_{NLP}(\mathbf{X}; \lambda), \mathbf{b}_{NLP}(\mathbf{X}; \lambda), \mathbf{t}_{0,NLP}(\mathbf{X}; \lambda))$ with the property that $\phi_{LP}(\mathbf{X}; 0) = \mathbf{X}$ while $\phi_{LP}(\mathbf{X}; 1) = \phi_0(\mathbf{X})$ and similarly for the

other load terms. Examples of nonlinear parametrizations are given later in the paper.

The only remaining issue is how to select the sequence of λ_k's. We use an adaptive rule defined as follows. Assume that there are no body forces and that the traction boundary conditions are all zero (i.e., "traction-free" surfaces). This means that the only loading term is the Dirichlet boundary condition. We form the deformed mesh M_{k-1} after applying the displacements given by $\mathbf{u}_{k-1}$ to non-Dirichlet nodes and Dirichlet boundary conditions scaled by λ_{k-1}, (i.e., the deformed position is given by $\mathbf{X} + \lambda_{k-1}(\phi_0(\mathbf{X}) - \mathbf{X})$ in the case of linear parametrization) to Dirichlet nodes. Next, we compute a value of λ_k such that, if the boundary nodes in M_{k-1} are further deformed to positions given by $\mathbf{X} + \lambda_k(\phi_0(\mathbf{X}) - \mathbf{X})$, then no tetrahedron altitude will decrease by more than a factor of η, where $\eta \leq 1$ is a tuning parameter of the continuation algorithm. (In particular, the step is sufficiently small that the mesh will not tangle after the new boundary condition given by λ_k is applied to M.) This adaptive strategy appears to work reasonably well, although we did encounter some robustness problems discussed in Section 7.

The description in the previous paragraph assumed the special case of traction-free Neumann boundaries and absence of body forces. It is more difficult to use this adaptive technique when there are nonzero body forces or tractions since it is not obvious how to step these loads in a way that prevents tangling on each step. Therefore, our test cases involve only traction-free, body-force-free loading conditions. Since the focus of the paper is the UBN method, it represents a strengthening of our contention that UBN is usually better than the competing algorithm (continuation) since we limit our testing only to the case that seems well suited for continuation.

4 Iterative stiffening for mesh untangling

In this section, we describe our procedure called iterative stiffening for untangling a mesh. We take the original mechanical problem given by (3), and using the same boundary conditions and loads, we solve the equations of isotropic linear elasticity using piecewise linear (constant-strain) finite elements [1]. Note that these equations have the same material parameters (the Lamé constants λ and μ) as the Mooney-Rivlin model. Linear elasticity requires one linear system solve. If the deformed mesh (i.e., the mesh that arises from moving the vertices to their displaced positions) is untan-

gled, then iterative stiffening is finished. If not, then our iterative stiffening procedure locates all elements that are inverted in the deformed mesh and increases their stiffness by 50%. The linear elasticity model is now solved again. This procedure is repeated indefinitely until the mesh is untangled or an excessive number of iterations has passed.

We have not found this precise version of iterative stiffening appearing in the previous literature, but it is related to ideas already in the literature. It could be regarded as an extension of FEMWARP, a finite element based mesh warping approach developed by the authors within the linear weighted laplacian smoothing (LWLS) framework [16]. One difference is that FEMWARP does not easily encompass the mechanical concept of traction boundary conditions. It is also related to a mesh warping method used for ALE solvers and described in Chapter 7 of [1].

In our preliminary version of the UBN method [17], the untangling was done using Opt-MS [4] rather than iterative stiffening. Opt-MS is an untangling algorithm that iteratively repositions interior nodes one at a time until the mesh is untangled. It solves a small linear-programming problem for each node to find the position for it that maximizes the minimum determinant of its neighbors. We found recently that iterative stiffening is more effective for use in UBN than Opt-MS. One possible reason is that it is difficult to implement traction boundary conditions in a natural way in Opt-MS.

Note that iterative stiffening can be made particularly efficient by using matrix-updating. In particular, it is well-known (see, e.g. [5]) how to update a Cholesky factorization of a symmetric positive definite matrix A after A has undergone a low-rank update. If the iterative stiffening procedure stiffens only a few elements per iteration (our test runs confirm that indeed there are usually only a few updates per step), then this can be implemented as a a low-rank update, which is potentially much more efficient than solving a new stiffness matrix from scratch. We did not implement matrix-updating because the work for iterative stiffening was usually dominated by the solver part of the algorithm anyway.

5 Newton Line Search

Newton's method is often employed for solving nonlinear systems of continuously differentiable equations [3]. Let $f : \mathbf{R}^{dm} \rightarrow \mathbf{R}^{dm}$, continuously differentiable, be given. The task at hand is to find a $\mathbf{u} \in \mathbf{R}^{dm}$ such that

$f(\mathbf{u}) = \mathbf{0}$. Let $\mathbf{u}_0 \in \mathbf{R}^{dm}$ be given. Then, at each iteration k, Newton's method solves

$$J(\mathbf{u}_k)\mathbf{s}_k = -f(\mathbf{u}_k), \tag{4}$$

where J denotes the Jacobian of f, for the Newton step, $\mathbf{s}_k$, and performs the following update

$$\mathbf{u}_{k+1} = \mathbf{u}_k + \mathbf{s}_k. \tag{5}$$

If it becomes necessary to satisfy one or more additional inequality constraints, it is possible to safeguard the Newton step with the introduction of a line search. Let α_k denote the line search parameter. Then α_k is chosen to be as large as possible such that $0 < \alpha_k \leq 1$ and $\mathbf{u}_{k+1} = \mathbf{u}_k + \alpha_k \mathbf{s}_k$ satisfies the constraint.

It is often difficult to compute the value of α_k that minimizes $f(\mathbf{x}_k + \alpha_k \mathbf{s}_k)$ and satisfies the constraint because f is often a highly nonlinear function. In addition, the optimal value of α_k often produces steplengths that are too short in practice. Thus, it is common practice in interior point methods to derive heuristics for computing α_k that allow for both ease of computation and larger steplengths [19]. One such heuristic is to choose α_k so as to stay a fixed percentage away from the boundary. We employ this heuristic in our line search below.

As was pointed out in Section 2, the mesh is tangled unless $J = \det(F) > 0$. Thus, we introduce a line search that enforces that $J > 0$ on each iteration of Newton's method. In particular, we begin with $J > 0$ on the zeroth iteration and choose the line search parameter α_k such that $J(\mathbf{u}_{k+1}) \geq 0.1 J(\mathbf{u}_k)$ on each element so as to stay a fixed percentage away from the boundary for reasons discussed above.

The following pseudocode algorithm shows how the line search parameter is determined. Let N denote the number of elements in the mesh. Given a displacement vector $\mathbf{u}$, it is straightforward for each element $i = 1, \ldots, N$ to compute the deformation gradient F and its determinant J determined by this displacement on element i; we denote the resulting determinant by $J(\mathbf{u}, i)$.

Let $\mathbf{u}_0$ be the value of the displacement (at non-Dirichlet nodes) returned by the previous iteration of our Newton/line search algorithm. Initially, the value of $\mathbf{u}_0$ is the output of the iterative stiffening algorithm. It is assumed that the mesh determined by the Dirichlet boundary conditions and by $\mathbf{u}_0$ on non-Dirichlet nodes is untangled. Let $\mathbf{s}$ denote the Newton step determined from $\mathbf{u}_0$ via (4).

$\alpha = 1$;
for $i = 1 : N$
    while true
        if $J(\mathbf{u}_0 + \alpha \mathbf{s}, i) \geq J(\mathbf{u}_0)/10$
            break
        end
        $\alpha = \alpha \cdot 0.9$;
    end
end

6 2D Experiments

We designed a series of numerical experiments in order to test the robustness of UBN and to compare it to the standard Newton continuation algorithm. For all of the numerical experiments in this paper, we set the parameters in (1) as follows: $\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}$ and $\mu = \frac{E}{2(1+\nu)}$, with $E = 1$ and $\nu = 0.3$.

The termination criteria for the Newton loop in UBN and for the final step of Newton continuation was that $\|F\|_2 \leq 10^{-10}\|F_0\|_2$, where F_0 is the initial value (i.e., the value when all interior displacements are set to 0) of the load vector. For the Newton continuation steps prior to the final step, the termination criteria was that $\|F\|_2 \leq 10^{-3}\|F_{k_i}\|_2$, where F_{k_i} is the initial value of the load vector at the beginning of major iteration k.

The algorithms were implemented in Matlab. The current version of the solver does not allow specification of nonzero traction boundary conditions nor specification of body forces. There is no difficulty in principle in extending the Matlab code for UBN to handle these types of loads, but new quadrature formulas would have to be written. As mentioned earlier, there are some conceptual difficulties in extending our adaptive method for selecting λ_k's for continuation to these types of loads.

The linear solution operation in Matlab is quite highly optimized and is expected to compete well with a custom-written C or C++ linear solver. On the other hand, the matrix assembly process involves several nested Matlab loops and is therefore expected to be much slower than a C or C++ version. For this reason, wall-clock times derived from the Matlab code are not useful predictors of computational demands that would be observed with a C or C++ code.

Instead, we measure the running time in terms of assembly/linear solve

steps. An assembly/linear solve (ALS) step consists of one stiffness matrix and load vector assembly operation followed by one sparse linear system solve. The Newton continuation method involves a sequence of Newton solve procedures, and each Newton solve is further subdivided into several ALS steps. The UBN method involves iterative stiffening iterations followed by a safeguarded Newton method. We count each iteration of iterative stiffening as an ALS step. The assembly portion of the iterative stiffening ALS operation is not exactly the same as the assembly portion of Newton, since the former involves linear elasticity assembly whereas the latter involves nonlinear tangent stiffness assembly. We ran both assembly codes on an older Windows machine running Matlab 5.3, which has a "flops" function built in that measures floating point operations. (Newer versions of Matlab lack this function.) From this experiment we determined that the number of operations for the two kinds of assembly are fairly close. Furthermore, both assembly operations are much less costly than the linear system solve. Note that the iterations of iterative stiffening would be considerably cheaper than an ALS step had we implemented low-rank corrections described in Section 4.

The solver portion of UBN involves additional operations connected with the line search. We determined (again by running test cases in Matlab 5.3) that the line search requires a tiny number of operations in comparison to the solution of the linear equations.

Thus, it is sensible to compare the running time of UBN to continuation by considering the total number of ALS steps required for either.

In this section we describe our 2-dimensional test case, which is an annular domain. The mesh was generated with Shewchuk's Triangle [15] and is illustrated in Fig. 1. It contains 182 nodes and 286 triangles.

The boundary conditions used in this test case involve a rotation of the exterior boundary circle by f radians combined with moving the inner boundary by a factor f closer to the outer boundary (where $f = 0$ means no motion and $f = 1$ means that the inner boundary would coincide with the outer boundary). Values of f tried were 0.1, 0.3, 0.6, and 0.7. The resulting deformed meshes are illustrated in Fig. 2.

The number of ALS steps to compute these deformed meshes is given in Table 1. The columns of this table are as follows. The first column f is the amount of boundary deformation as described in the previous paragraph. The second column UBN–IS is the number of iterations of iterative stiffening required by UBN. The third column UBN–NM is the number of iterations of Newton's method required by UBN. The fourth column UBN-ALS is the

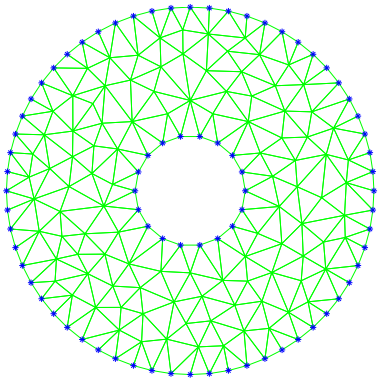

Figure 1: Annulus used for testing in this section.

number of ALS steps required by UBN (and hence is the sum of the second and third columns).

The remaining columns of the table report on results from the continuation algorithm. The fifth column is the number of major iterations (i.e., updates to the continuation parameter λ) required by the continuation algorithm using the adaptive rule discussed in Section 3 with parameter $\eta = 1/3$. The sixth column is the number of ALS steps required by continuation. The seventh and eighth columns are the same quantities when $\eta = 1.2$. Note that $\eta = 1.2$ is a quite aggressive choice of stepsize for continuation, since any value of $\eta > 1$ means that updating the boundary could cause an inversion. For this 2D test case, an aggressive choice of η did not seem to hinder convergence, but the results for a large value of η in 3D described in the next section are less favorable.

For $f = 0.8$, neither method was able to find a solution. UBN's iterative stiffening did not untangle the mesh after the maximum number of iterations (400) had been reached, and continuation stalled at $\lambda = 0.93$.

It should be noted that a highly deformed mesh like the solution when $f = 0.7$ is probably not physically valid because the finite element discretization is no longer an accurate approximation to the underlying PDE. Nonetheless, we include extreme cases like this because it is interesting to compare the two algorithms in limiting cases. The test results show that UBN is much faster than continuation for both modest and extreme deformations.

Note that for continuation, the outer boundary motion is parametrized by θ, the rotation angle. Linear parametrization with respect to (x, y) co-

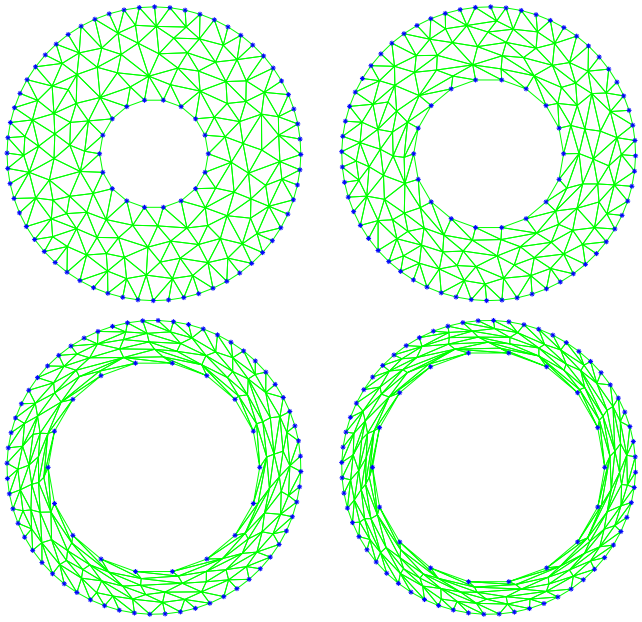

Figure 2: Deformed annulus meshes with $f = 0.1$, $f = 0.3$, $f = 0.6$, and $f = 0.7$ respectively.

ordinates would work poorly in this case because a linear deformation from the initial position of the outer boundary to the final position would cause the outer boundary to shrink in radius and then expand.

Comparing the UBN–ALS and Contin–ALS columns of this table indicates that UBN is approximately 15 to 30 times more efficient than continuation when $\eta = 1/3$. Continuation is 3 to 4 times faster when used with the larger value of η but is still significantly slower than UBN. Other annulus deformation tests not reported here confirm that UBN is always far more efficient than continuation.

We also wished to check whether the line search procedure built into UBN was ever active in order to determine whether it is an essential part of UBN. We found that it was active on about 30%-50% of the iterations for the larger values of deformation.

Table 1: Results of comparison of UBN to continuation for the 2D annular domain. See the text for meanings of the columns.

f	UBN			Continuation			
				$\eta = 1/3$		$\eta = 1.2$	
	IS	NM	ALS	MajIt	ALS	MajIt	ALS
0.1	1	3	4	28	57	8	18
0.3	1	5	6	87	175	24	50
0.6	5	20	25	265	483	73	148
0.7	9	30	39	414	683	113	228

7 3D Experiments

Our experiments in 3D consisted of two tetrahedral meshes called "Hook" and "Foam5," which were provided to us by P. Knupp [9]. The sizes of the meshes are as follows: Hook contains 1190 vertices and 4675 tetrahedra, and Foam5 contains 1337 nodes and 4847 tetrahedra. Hook is contained in a bounding box of size $54 \times 40 \times 95$, while Foam5 is contained in a bounding box of size $11.3 \times 5.5 \times 6.6$.

In both cases, we applied Dirichlet boundary conditions to two of the boundary surfaces, leaving the rest traction-free. In both cases, the Dirichlet conditions are identically zero on one boundary surface and are pulling in a uniform direction on the other surface by three magnitudes. For Hook, the displacement sizes were 10, 20, and 40, whereas for foam they were 0.5, 2, and 5. Thus, we see that the applied displacements are on the same order as the size of the object, and therefore large deformations will result. Figure 3 shows the deformed and undeformed configurations of Hook for the maximum deformation of 5, while Fig. 4 shows the corresponding illustration of Foam5.

The results of our tests of UBN versus Newton continuation on Hook and Foam5 are given in Table 2. As in the previous section, the unit for measuring running time is ALS steps. For these tests, straight linear parametrization was used for continuation. As before, our results on Hook indicate that UBN is 15 to 50 times faster than continuation when $\eta = 1/3$, and 5 to 15 times faster for $\eta = 1.2$.

The continuation algorithm terminates when the increment in λ became smaller than the prespecified minimum (0.0005); this happened in two of the

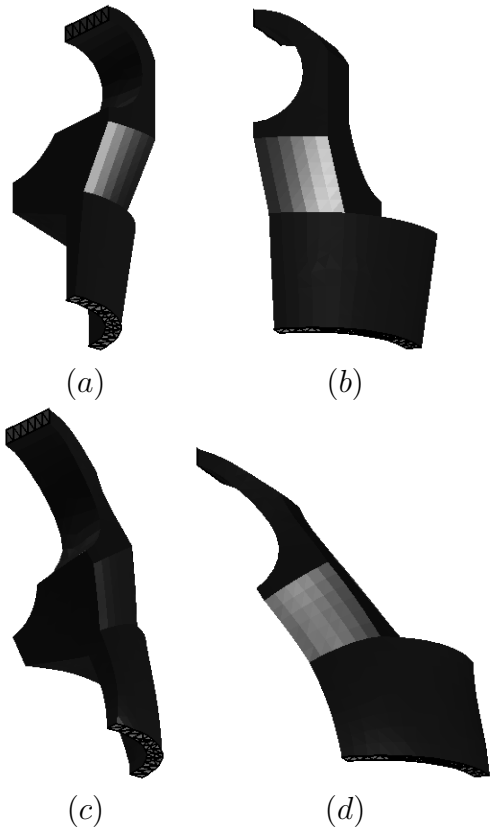

Figure 3: The top row (a) and (b) shows the undeformed Hook mesh from two different viewpoints. The dark blue surface is traction free, while the Dirichlet boundary conditions are applied to the green and pink surfaces. The bottom row (c) and (d) show Hook after the maximum deformation of 40 is applied.

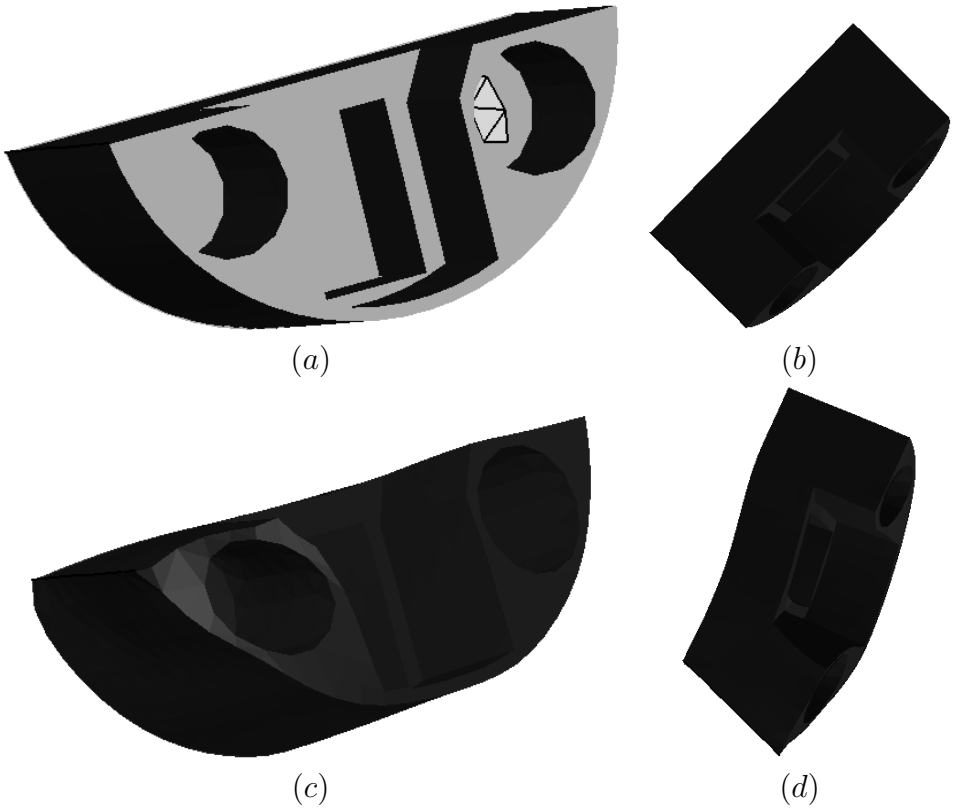

Figure 4: The top row (a) and (b) shows the undeformed Foam5 mesh from two different viewpoints. The dark blue surface is traction free, while the Dirichlet boundary conditions are applied to the green surface and pink edge. The bottom row (c) and (d) show Foam5 after the maximum deformation of 5 is applied.

Table 2: Results of comparison of UBN to continuation for 3D domains. See the text for meanings of the columns.

mesh	applied displ.	UBN			Continuation			
					$\eta = 1/3$		$\eta = 1.2$	
		IS	NM	ALS	MajIt	ALS	MajIt	ALS
Hook	10.0	1	5	6	53	107	15	32
Hook	20.0	1	6	7	105	212	30	61
Hook	40.0	1	8	9	210	421	59	119
Foam5	0.5	1	4	5	28	36	***	***
Foam5	2.0	1	5	6	—	—	***	***
Foam5	5.0	1	7	8	—	—	***	***

tests with Foam5 when $\eta = 1/3$ (indicated by '—' in the table). Apparently this is due to an extremely flat tetrahedron which, although it is does not become inverted, causes the heuristic used for adaptively incrementing λ to take very conservative steps.

The continuation algorithm also terminates when inverted elements remain after one major iteration. This happened in three of the tests with Foam5 when $\eta = 1.2$ as indicated by '***' in the table. This shows that, as expected, the aggressive choice of η may be more prone to inverting elements.

8 Conclusions

In summary, we developed a robust solution method for solving nonlinear elasticity equations for hyperelastic solids with large boundary deformations. The basic idea is to first untangle the mesh using purely geometric methods and second solve the mechanical model; thus, the algorithm was named UBN (for "untangling before Newton"). The first step of our algorithm is to attempt to untangle the mesh with iterative stiffening. Assuming the mesh is untangled, UBN takes safeguarded Newton steps to solve (3).

We tested the robustness of UBN and compared it to the standard Newton continuation algorithm. We demonstrated that UBN is much faster than the Newton continuation algorithm. It could be argued that continuation would be more competitive with UBN if only we had used a different strat-

egy for incrementing λ_k. This may be true, but it seems to us that there is no good universal fast method for choosing the λ_k. Our experiments indicate, for example, that a more aggressive algorithm for updating λ is more prone to terminating early due to inverted elements. Even selecting the continuation path seems to be nontrivial (e.g., for the 2D annulus example, it was necessary to parametrize the Dirichlet boundary condition in polar rather than rectangular coordinates). In contrast, the UBN method does not require any such analogous problem-dependent decisions, and the only parameters of the algorithm involve termination criteria.

As described so far, our method applies to finite elements in which the displacement field is piecewise linear over tetrahedra, but UBN could be extended to piecewise quadratic displacements. The challenge with piecewise quadratic displacements is that checking for tangling is much more complicated, as J is not constant on the element. In particular, it is a function of both the displacement and the location on the element, which makes it difficult to determine when $J > 0$ analytically. There are some separate necessary and sufficient conditions for element inversion in the literature. Let G be the Jacobian of F. Then one such necessary condition is that $\det(G)$ has the same sign (strictly positive or strictly negative) at some finite list of test points [14]. In this case, we would test for inversion at the Gauss points used for numerical quadrature; however, it is still possible that folding could occur at the corners. A more complicated sufficient condition for invertibility involving the Bernstein-Bézier form of a polynomial is given in [18]. Checking that the sufficient condition is met requires running a linear programming algorithm. Salem, Canann, and Saigal have proposed sufficient conditions for quadratic triangles and tetrahedra in [10], [11], [12], and [13].

Finally, it should be noted that there are some classes of problems that are not amenable to UBN. For example, consider a long cylinder in which one end is held at zero displacement, the other is rotated by 2π radians, the long side-surface is traction-free, and there are no body forces. For this problem, UBN would compute a solution where all displacements are identically zero.

The difficulty with this problem instance lies in the fact that for this geometry and specification of boundary conditions, there are an infinite number of continuum solutions (namely, rotation of one endcap by any integer multiple of 2π). To distinguish one solution from another requires additional information beyond boundary conditions. It is not clear what form that additional information ought to take. (For this particular problem instance, the additional data is the particular sought-after multiple of 2π, but it is not

clear what specification is needed in general.) It is also not clear how UBN should be extended in order to use such additional information.

9 Acknowledgements

The authors wish to thank Patrick Knupp of Sandia National Laboratories for providing us with the 3D test meshes. They benefited from many helpful conversations with Katerina Papoulia of Cornell University.

References

[1] T. Belytschko, W. Liu, and B. Moran. *Nonlinear Finite Elements for Continua and Structures*. Wiley, 2000.

[2] P. Ciarlet and G. Geymonat. Sur les lois de comportement en élasticité non-linéaire compressible. *C. R. Acad. Sci. Paris Sér. II*, 295:423–426, 1982.

[3] J. E. Dennis, Jr. and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Society for Industrial and Applied Mathematics, 1996.

[4] L. Freitag and P. Plassmann. Local optimization-based simplicial mesh untangling and improvement. *Int. J. Numer. Meth. Eng.*, 49:109–125, 2000.

[5] G. H. Golub and C. F. Van Loan. *Matrix Computations*. John Hopkins University Press, 3rd edition, 1996.

[6] G. Holzapfel. *Nonlinear Solid Mechanics: A Continuum Approach for Engineering*. Wiley, 2000.

[7] T. J. R. Hughes. *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*. Dover Publishers, 2000.

[8] P. Keast. Moderate-degree tetrahedral quadrature formulas. *Comput. Method. Appl. M.*, 55:339–48, 1986.

[9] P. Knupp. Personal Communication, October 2002.

[10] A. Salem, S. Canann, and S. Saigal. Robust quality metric for quadratic triangular 2D finite elements. *Trends in Unstructured Mesh Generation*, 220:73–80, 1997.

[11] A. Salem, S. Canann, and S. Saigal. Mid-node admissible spaces for quadratic triangular 2D arbitrarily curved finite elements. *Int. J. Numer. Meth. Eng.*, 50:253–272, 2001.

[12] A. Salem, S. Canann, and S. Saigal. Mid-node admissible spaces for quadratic triangular 2D finite elements with one edge curved. *Int. J. Numer. Meth. Eng.*, 50:181–197, 2001.

[13] A. Salem, S. Saigal, and S. Canann. Mid-node admissible space for 3D quadratic tetrahedral finite elements. *Eng. Comput.*, 17:39–54, 2001.

[14] M. Shephard, S. Dey, and M. Georges. Automatic meshing of curved three-dimensional domains: Curving finite elements and curvature-based mesh control. In I. Babuska, J. Flaherty, J. Hopcroft, W. Henshaw, J. Oliger, and T. Tezduyar, editors, *Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations*. Springer Verlag, 1995.

[15] J. Shewchuk. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Proceedings of the First Workshop on Applied Computational Geometry*, pages 124–133, New York, NY, 1996. Association for Computing Machinery.

[16] S. Shontz and S. Vavasis. An algorithm based on finite element weights for warping tetrahedral meshes. Archived by arxiv.org, cs.NA/0410045, 2004.

[17] S. M. Shontz. *Numerical Methods for Problems with Moving Meshes*. PhD thesis, Cornell University, 2005.

[18] S. Vavasis. A Bernstein-Bézier sufficient condition for invertibility of polynomial mappings. Archived by arxiv.org, cs.NA/0308021, 2003.

[19] S. J. Wright. *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics, 1997.